BMD Agents: An Agent-Based Framework to Model Ballistic Missile Defense Strategies

Duminda Wijesekera[1], James B. Michael[2,*], Anil Nerode[3]

[1]*George Mason University,Fairfax, Va., U.S.A. dwijesek@gmu.edu*
[2]*Naval Postgraduate School, Monterey, Calif., U.S.A. bmichael@nps.edu*
[3]*Cornell University, Ithaca, N.Y., U.S.A. anil@math.cornell.edu*

*** Corresponding author**

Paper no. 339

Report Documentation Page

Form Approved
OMB No. 0704-0188

1. REPORT DATE	2. REPORT TYPE	3. DATES COVERED
JUN 2005		**00-00-2005 to 00-00-2005**

4. TITLE AND SUBTITLE	5a. CONTRACT NUMBER
BMD Agents: An Agent-Based Framework to Model Ballistic Missile Defense Strategies	5b. GRANT NUMBER
	5c. PROGRAM ELEMENT NUMBER
6. AUTHOR(S)	5d. PROJECT NUMBER
	5e. TASK NUMBER
	5f. WORK UNIT NUMBER
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) **George Mason University,4400 University Drive,Fairfax,VA,22030**	8. PERFORMING ORGANIZATION REPORT NUMBER
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)	10. SPONSOR/MONITOR'S ACRONYM(S)
	11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release; distribution unlimited

13. SUPPLEMENTARY NOTES
The original document contains color images.

14. ABSTRACT

15. SUBJECT TERMS

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT **unclassified**	b. ABSTRACT **unclassified**	c. THIS PAGE **unclassified**		**37**	

BMD Agents: An Agent-Based Framework to Model Ballistic Missile Defense Strategies

Duminda Wijesekera
ISE Department and CSIS
George Mason University
Fairfax, VA 22030
dwijesek@gmu.edu

James Bret Michael
Computer Sci. Department
Naval Postgraduate School
Monterey, CA 93943
bmichael@nps.edu

Anil Nerode
Mathematics Department
Cornell University
Ithaca, NY 14853-7901
anil@math.cornell.edu

ABSTRACT

We introduce a model-based methodology for comparative evaluation of the effectiveness of alternative ballistic missle defense strategies. The major new feature is that BMD is modelled as a distributed system of interacting agents in which some agents are physical (such as sensors and launch systems) and some are rule-based (such as decision makers and threat-evaluators). In this model, doctrine, policy and organizational structures are treated as rule-based constraints on system behavior. This will allow both analytical and simulation modeling of alternative BDM strategies based on varying scenarios. Alternative proposals can be evaluated side by side by analysis and simulation of the effects of putting in and taking out physical and rule based agents and constraints for alternative sensors, launch systems, threat-evaluation systems, command and control systems, and doctrine. The effects of alternative organization can be evaluated. We demonstrate the use of the methodology for evaluating three candidate command structures: hierarchical, partially flattened, and flattened.

Key Phrases: Agents, Ballistic Missile Defense, Distributed Decision Architecture, Organizational Structure, Policy.

1. INTRODUCTION

Strict time budgets imposed on decision making in the context of defending against attack from modern ballistic missiles have rendered the traditional command structure, in which many humans are placed in decision loops that rely on serial communication along the decision making chain, impractical. The upper bounds on decision-times for assigning a weapon to intercept a threat object ranges from approximately thirty seconds for short-range ballistic missiles to a few tens of minutes for long-range ballistic missiles. In general, the likelihood of intercepting a ballistic missile decreases as the missile nears its target. For instance, in the terminal phase of flight, the threat missile can deploy difficult-to-defeat countermeasures. Even if the interceptor destroys the ballistic missile and its warhead, the second-order effects of the kill would pose a danger to civilians in whatever area the warhead remnants and missile debris would fall. Intercepting a ballistic missile in the midcourse of its flight could be problematic, as hundreds of small bomblets could be released from the ICBM just after its ascent. Because these bomblets would strike separately, no known midcourse or terminal defense can halt such an attack [6].

However, there is a lower bound on the decision-time to launch a weapon, because a commander cannot assign a weapon to engage an object until it has been classified as a threat missile. Further, the commander may have to receive weapon-release authority from higher-ups in the chain of command, coordinate the use of available weapons with other commanders, and plan crisis-action before assigning a weapon. The key point here is that the the sum of the sensor-processing, communication, weapon-assignment, and actuation (i.e., release of the weapon) delays, can be greater than the total time than that the threat missile is in its boost phase. Therefore, it may not be possible to intercept the threat missile in the boost phase with today's technology, although this situation may change with technical advances that may come in the near future [7]. According to Athans [1] the effectiveness of the tactical decisions impact all reasonable measures of effectiveness (MoE), such as:

1. Do interceptors destroy threat missiles?
2. Do follow-up shots destroy missed missiles?
3. Does the system hold fire when an object is reclassified as benign?

Similarly, examples of measures of performance (MoP) include:

1. Do interceptors destory threat missiles at sufficiently high altitude to negate the effect of the warheads, payload (e.g. chemical weapons of mass destruction)?
2. Do launched weapons have time to engage threat missiles?
3. Can the system discriminate between benign objects and threat missiles?

This research was funded in part by a grant from the U.S. Missile Defense Agency. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of the U.S. Government. The U.S. Government is authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright annotations thereon.
The 10th Command and Control Technology Symposium, 2005 June 13-16, McLean, VA

One means to address such measures of effectiveness and performance is to investigate means to improve the effectiveness and performance of the organizational structure and processes that support decisions made by the BMD system. A command structure needs to provide for unity of command, unity of effort, a common operating picture, and situational awareness. In order to ensure these properties, it is necessary to identify the information-flow requirements and decision-making aids to permit the commanders to issue orders in a timely manner. Weller et al. [13] propose two alternative command structures to the *status quo*: a compressed *chains of command (CoC)* for regional commanders and a flattened CoC for BMD. The motivation for flattening the command structure is to reduce the decision-time. Weller et al. propose that speed, flexibility, and situational awareness be used to compare and contrast the alternative command structures. However, Athans [2] noted that there is no systematic methodology to do so.

We present a methodology for systematically modeling organizational structures that assess the MoEs and MoPs due to the cause-and-effect relationships inherent in distributed decision making. We model them as a distributed team of decision-making agents. Doctrine and policy, modeled in the form of rules, constrain the behavior of agents within the system. Traditionally policy is considered a choice of actions among a set of lawful alternatives and doctrines is recognized as a means by which the chosen policy will be effectuated. We demonstrate the use of the methodology to evaluate three candidate command structures for global BMD: hierarchical, partially flattened, and flattened.

Section 2 of this paper describes basic requirments for C2 models. Section 3 presents the formal syntax of our agents. Section 4 describes our syntax to combine them in forming organizational structures for C2 systems. Section 5 presents our analysis and section 6 presents related work. Section 7 concludes the paper.

2. MODELING REQUIREMENTS

BMD planning both deliberate and crisis-action planning. *Deliberate planning*, involves creating templates for prosecuting missile engagements. *Crisis-action planning*. The parameters described in the templates are filled in when a situtation occurs and instantiated within the components of the BMD system. Crisis-action planning is used to modify plans as a battle progresses. However crisis-planning difficult to perform during a missile engagement because the decision time budgets are short. The planners follow a conceptual operational paradigm referred to as a *kill chain* consisting of surveillance, detection, tracking, identification of targets, weapons assignment, engagement, and kill assessment [8, 3]. The execution of tasks within the kill chain are geographically distributed, where surveillance requires a geograhically dispersed collection of sensors, referred to as *sensor networks*. Detection requires collecting information from many types of sensors and track processing may require equipment capable of continuously tracking flight trajectories. Weapons assignment may be the responsibility of a commanding officer. Engagement of the threat missile may be allocated to a commander in a different theater or region than that from which the interceptor was launched. Kill assessment may be performed across regions or theaters. Thus, the kill chain is cooperatively executed by a collection of entities that are tasked to carry out some parts of the overall objective: to destroy threatening missiles.

Entities that participate in a kill-chain are legal combatants, goverened by individually assigned duties and policies. Therefore, we refine the principle objective of this paper to develop a framework to model these duties of individuals that participate in kill chains designed for ballistic missile defense and to be able to measure their effectiveness and performance. We intend to bring the formal tools that are used to model and analyze policies in distributed systems to one of the imporatant defense objectives. In order to do so, we propose a framework that provides the building blocks to specify the duties of the entities that execute kill chains as using *event-condition-action rules (ECA)*, similar to those used in agent-based systems. We then show how our framework can be used to construct different command structures that represent policies adpoted by military organizations. We then show how to analyze their effectiveness in achieving the eventual military objectives that they are designed to fulfil. The next section introduces the formal syntax of our modeling language.

3. FORMAL SYNTAX

Because the duty cycles of our BMD agents are formalized as ECA rules, we first define their formal syntax and show how to build agents that can execute tasks in a BMD kill chain. We compose these agents to build appropriate C2 structures.

DEFINITION 1 (EVENTS). *$\mathcal{E} = \{e_i : i \geq 1\}$ is a set of predicates of finite arities, referred to as events. The values of their attributes are chosen from a finite collection of domains, say $DOM_1, \ldots, DOM_n$.*

For example, `detected(flyObj,lat30`^{0}`,long28.5`^{0}`, GMT11.20.37)` is our symbol for the event that a flying object was detected at latitude 30^0 and longitude 28.5^0 at time GMT 11.20.37. We now define the syntax for the conditions.

DEFINITION 2 (CONDITIONS). *Let $\mathcal{C} = \{c_i : i \geq 1\}$ be a collection of predicates of various arities disjoint from $\mathcal{E}$, whose attribute values come from domains $DOM_1, \ldots, DOM_m$ for some $m \geq n$. We say that these constitute atomic conditions. We recursively define conditions as:*
1. An atomic condition is a condition.
2. If c_1 and c_2 are conditions, then so are $c_1 \wedge c_2, c_1 \vee c_2, \neg c_1$.

For example, `shortRangeStockpile(Country-A, 1000)` $\wedge$ `status(atWar,Country-A,Country-B)` is a condition that says that there are 1000 short range countermissiles in country-A and that it is at war with Country-B. We now define ECA rules that specify how our BMD agents react to events under a given set conditions. In order to do so, we first define communication and execution primitives used by them.

DEFINITION 3 (ADDRESSES AND ACTIONS). *A domain DOM_0 is the address space. An address term is either an address constant (i.e., an element from DOM_0) or a variable.*

send(E,$Y_s, Y_r, t, \vec{z}$): *is referred to as a **send primitive** where `E` is an event and Y_s and Y_r are address terms. t is a non-negative real number. `send(E,`Y_s, Y_r, t`)` is our notation for an agent Y_s informing an(other) agent Y_r of an occurance of the event `E`. $\vec{z}$ contains any other data used in the event. The message is enqued for delivery at local time t.*

recv(E,$Y_s, Y_r, t, \vec{z}$): *is referred to as a **receive primitive** where* `E` *is an event and Y_s and Y_r are address terms and t is time.* `recv(E,`Y_s, Y_r, t`)` *is the notation for an agent Y_r receiving event* `E` *sent by an agent Y_r at local time t. $\vec{z}$ is the same as before.*

exec($f, \vec{z}$): *is referred to as an **execute primitive** where f is a function call with parameters $\vec{z}$ - our notation for executing $f(\vec{z})$ within a BMD agent.*

Definition 3 has primitives that can be used by an agent Y_s to inform another agent Y_r about an occurance of an event `E`. The time of occurrences of `E`, the time of sending t by Y_s (i.e., t in send(E,Y_s, Y_r, t)), and the time of receipt of `E` at Y_r (i.e., t in recv(E,Y_s, Y_r, t)) are all measured with respect to local times of the involved agents and do not have to be equal. Consequently, our framework makes no assumptions about synchronized clocks. We now use these communication messages in rules that specify agent behavior.

DEFINITION 4 (EVENT-CONDITION-ACTION RULES). *An event-action condition rule (ECA rule) is of the form*

$$If(recv(E, Y_s, Y_r, t, \vec{z}))\ and\ (C)\ then\ \phi$$

or of the form

$$If(C)\ then\ \phi$$

where:

1. `E` *is an event and Y_s, Y_r are address terms.*
2. `C` *is a condition.*
3. ϕ *is either* `exec(`$f, \vec{z}$`)` *or* `send(E,`$Y_s, Y_r, t, \vec{z}$`)` *where* `E` *is an event and Y_s is the sender's address, Y_r is the receiver's address, t is the current local time of the sending agent and $\vec{z}$ contains auxillary data.*

We use the notational convention that an agent refers to its own address as `self`. Although our syntax can be used to specify many agent behaviors, we define three kinds of agents: *theater commander agents (TCAs), regional commander agents (RCAs)* and *strategic commander agents (SCAs),* reflecting the C2 structures used in BMD management. These agents exibit the following common behaviors.

1. Agents operate under three modes: peacetime, pre-hostilities and hostilities, and when appropriate, change their own mode and call for others to do so.
2. Agents obtain sensory cues from and (where appropriate) update sensory information to a virtual address referred to as the `sensorNet`.
3. Agents obatain operational status of weapons from a virtual address, and if appropriate, update information about them, referred to as the `weaponsNet`.

Consequently, depending on the received commands or their own decisions (perhaps based on local computations), they change modes and pass selected changes of sensed data or weapons states to the appropriate virtual entities. We now specify the behavior of the aforementioned agents.

Policy 1: TCA(id,mySuperiors,sensorNet,weaponsNet)

Initialize `mySuperiors, weaponsNet, sensorNet`
Input: from `mySuperiors, sensorNet, weaponsNet`
Output: to `sensorNet, weaponsNet`
Data: `mode, cancelable[], nonCancelable[]`
/*$\vec{z}$ contains the position (x, y, z), velocity (v_x, v_y, v_z), acceleration (a_x, a_y, a_z) at time t etc that are used to compute possible position and projected flight trajectory, etc. */
while $recv(E, Y_s, Y_r, t, \vec{z})$ **do**
  switch *condition* **do**
    case *(Y_s =local sensor)*
      send(s,`self`,`sensorNet`,$\vec{z}$,time)
    case *(E =switchMode)*
      exec(switchMode,newMode)
    case *(E =launch and $Y_s \in$ `mySuperiors` and $Y_r =$ `self`)*
      for *j=1 to n* **do**
        if *exec(withinHit,threat,*$L_{self,j}, Lt_{self,j}$*)* **then**
          assignTargetToMB($L_{self,j}, m_{self,j}$)
          /*Assume Jth MBA can do Jth job */
          send(assigned,`self, weaponsNet`, now,$m_{self,j}, L_{self,j}, Lt_{self,j}$)
          exec(lock,$m_{self,j}, L_{self,j,Lt_{self,j}}$)
          send(locked,`weaponsNet, self`, now,$m_j, L_{self,j}, Lt_{self,j}$)
        else
          send(cantHit,self,Y_s,now,$L_{self,j}$)
    case *(E=enterNonCancelPhase,*$m_{self,j}$*)*
      send(enterNC,`weaponsNet, self`,now,$m_{self,j}$)
      exec(proceed,$m_{self,j}$)
    case *(E =cancel and $Y_s \in$ `mySuperiors` and $Y_r =$ `self`)*
      for *(j=1 to n)* **do**
        identifyMBA($C_{self,j}, m_{self,j}$)
        /*assume Jth job for Jth MBA */
        if *cancelable(*$m_{self,j}$*)* **then**
          exec(cancelLaunch,$m_{self,j}$)
          send(cancel,`weaponsNet, self`,now,$m_{self,j}$)
        else
          send(cantCancel,self,Y_s,now,$m_{self,j}$)
    case *($E = hold, m_{self,(j=1,n}$ by $d_{self,j}$, $Y_s \in$ `mySuperiors` and $Y_r =$ `self`)*
      for *(j=1,n)* **do**
        identifyMB($H_{self,j}, m_{self,j}$)
        /*Assume Jth job for Jth MBA */
        if *holdable(*$m_{self,j}$*)* **then**
          $hTimer_{self,j} = now + h_{self,j}$
          exec(setHoldTimer,$m_{self,j}, \vec{z}, d_{self,j}$)
          send(hold,`weaponsNet, self`,now,$m_{self,j}$)
    case *($E = hTimer, now = hTimer_{self,j}$)*
      exec(reLaunch,$m_{self,j}, \vec{z}$,now)
      send(reLaunch,`weaponsNet, self`,now,$m_{self,j}$)
    case *($E = reLaunch, m_{self,j}$, $Y_s \in$ `mySuperiors` and $Y_r =$ `self`)*
      if *delayed(*$m_{self,j}$*)* **then**
        exec(reLaunch,$m_{self,j}$,now)
        send(reLaunch,`weaponsNet, self`,now,$m_{self,j}$)
  send(weaponsUpdate,`weaponsNet, self`,Y_s,now,$\vec{z}$)
  send(sensorUpdate,`sensorNet, self`,Y_s,now,$\vec{z}$)

3.1 Theater Commander Agents

Theater commander agents are those that operate the missiles. In addition to the generic behavior, we assume that they work in a *duty cycle* that consists of:

1. Aquire a target and lock on to it.
2. Launch a missile. If asked and if is possible, delay or intercept (i.e., hold fire).
3. Assess the success of launched weapons (i.e., access kill).

Policy 1 specifies the behavior of TCA agents. As stated, TCAs perform a duty cycle consisting of executing commands sent by superiors (referred to as `mySuperiors` in Policy 1) and updating the status of sensors and weapons under its command to the `sensorNet` and `weaponsNet`, respectively. The commands expected from superiors are launching, canceling, delaying and relaunching weapons.

In order to specify the TCA behavior, Policy 1 uses several auxiliary function calls. Because military forces function differently under peacetime, preparing for hostilities and during hostilities, we model three *modes* of operation *peace*, *preWar* and *war* for our agents. The procedure `switchMode` switches the modes between `peace, preWar` and `war`. The procedures `lock, launch` locks a missile to a target and launches a missile towards a target. Similarly, `withinTarget, cancelable` returns a binary value indicating if a target is within reach of its missile and is in a position to cancel the launching process. The function `delayable` returns true if the missile launching is delayable and the command `delay` postpones the launching by a specified amount of time. `reLaunch` launches an already delayed launch. The two functions `assignTargetToMB` and `identifyMB` assign a launch request $L_{self,j}$ to a missile battery $m_{self,j}$ and identify a cancel or hold requests (given by $C_{self,j}$ or $H_{self,j}$ respectively) to the missile battery $m_{self,j}$ originally assigned to launch it, respectively. Although the latter is a table lookup, the former solves a resource allocation problem. Implementation of these functions are device specific, but their behavior satisfies the specification, and reflects BMD policies.

As specified in Policy 1, TCAs receive instructions to launch, delay or launching of a collection of missiles as a part of a regional objective. Based on the request and its own calculations, the receiving TCA allocates a local missile battery to launch the requested missile. If possible delaying and caneling requests are also honored. Due to the time taken for a missile to be dispatched, at any given time there could be many missiles that are on their way from the TCA to their targets. The last two steps of the duty cycle update the `weaponsNet` and `sensorNet` of the status of their own weapons systems and sensory devices.

3.2 Regional Commander Agents

Policy 2 specifies the behavior of regional commander agents (RCAs). An RCA coordinates the activities of the TCAs under its command. In this process, they are handed down *regional military objectives* that have to be met by devising and executing an engagement plan using its TCAs. That may include launching many missiles at different times, (such as firing a second missile if the first fails to destroy or damage the targets). Sometimes, RCAs may not have superiors so that they have to decide when to initiate a regional operation.

Policy 2 uses several auxiliary functions and procedures. The function `startRegional` returns true when a superiorless RCA is to start its own battle plan. The procedures `computeBattlePlan` ($\vec{z}, \vec{w}$, `canAchieve`) computes a battle plan described by the attribute vector $\vec{w}$ from a regional military objective described by the vector of parameters $\vec{z}$. `canAchieve` evaluates to true if and only if the requested plan can be achieved with resources available to the RCA. Policy 2 assumes that only one plan comes back as achievable. In the more general case, there could be more than one plan comming back multiple plans that are concurrently achivable. Our ongoing work addresses this requirment. Commands `newObjective, chgObjective` and `cnclObjective` are issued by SCAs to begin a new, change or cancel a regional objective respectively. If requested objectives cannot be met, then a RCA sends the message `cantAchieve` back to its superiors to inform its inability to fulfil the request with exisiting resources. If the objective can be met, the RCA computes a battle plan that consists of launching, delaying and cancelling existing schedules for TCAs under its command and informs them appropriately, using the following functions.

Policy 2: RCA(id,mySuperiors,myTCAs,sensorNet,weaponsNet)

Initialization: Get addresses of `mySuperiors, myTCAs, sensorNet, weaponsNet`
Input: Commands from `mySuperiors`, information from `sensorNet` and `weaponsNet`
Output: commands to `myTCAs`
while $recv(E, Y_s, Y_r, t, \vec{z})$ *or mode=war* **do**
  switch *condition* **do**
    case *(E =switchMode)*
      exec(switchMode,newMode,now)
    case *(E =newObjective $\vee E$ =chgObjective$\vee E$ =cnclObjective) and t =now and $Y_s \in$ `mySuperiors` and Y_r =self] or [`mySuperiors`$= \emptyset \wedge$ startRegional])*
      computeBattlePlan($\vec{z}, \vec{w}$,canAchieve)
      if *(canAchieve)* **then**
        computeFiringMBAandTime($\vec{w}$, $\overrightarrow{(TCA_i, \vec{L_{i,j}}, \vec{t_{i,j}})}$)
        computeHoldingMBAandTime($\vec{w}$, $\overrightarrow{(TCA_i, \vec{H_{i,j}}, \vec{t_{i,j}})}$)
        computeCancelMBAandTime($\vec{w}$, $\overrightarrow{(TCA_i, \vec{C_{i,j}}, \vec{t_{i,j}})}$)
        if E =cnclObjective **then**
          cancelBattlePlan()
        for *i=1 to n* **do**
          send(launch,self,$TCA_i, \vec{L_{i,j}}, \vec{Lt_{i,j}}$,now)
          send(holdFire,self,$TCA_i, \vec{H_{i,j}}, \vec{Ht_{i,j}}$,now)
          send(cancel,self,$TCA_i, \vec{C_{i,j}}, \vec{Ct_{i,j}}$,now)
      else
        send(cantAchieve,self,Y_s,now,$\vec{z}$)

computeFiringMBAandTime($\vec{w}$,$\overrightarrow{(TCA_i, \vec{L_{i,j}}, \vec{t_{i,j}})}$): Given a *high level* objective characterized using the attribute

vector $\vec{w}$ for TCA_i that is under the command of the given RCA, this procedure determines that the missile $L_{i,j}$ to be launched at time $t_{i,j}$.

computeHoldingMBAandTime($\vec{w}$,$\overrightarrow{(TCA_i, H_{i,j}, t_{i,j})}$): Given a *high level* objective characterized by the attribute vector $\vec{w}$ for TCA_i that is under the command of the given RCA, this procedure determines that the missile $H_{i,j}$ needs to be delayed by a time of $t_{i,j}$.

computeCancelMBAandTime($\vec{w}$,$\overrightarrow{(TCA_i, C_{i,j}, t_{i,j})}$): Given a *high level* objective characterized by the attribute vector $\vec{w}$ for TCA_i that is under the command of the given RCA, this procedure determines that launching missile $C_{i,j}$ needs to be cancelled at time $t_{i,j}$.

After computing the schedules, the RCA sends them to the appropriate TCAs accordingly. As will be seen in section 4, (atypically) under an altered C2 structure, an RCA may be required to function without a superior.

3.3 Strategic Commander Agents

SCAs model global strategy planners such as US STRATCOM or NATO commanders that coordinate many regional crises simultaneously. In order to specify such behaviors, SCAs use many procedures. Based on received information, `computeMode` computes the new operational mode from the old one. `toColleagues` compute the information conveyed to their colleagues. The procedure `regionalObjective` computes the collection of regional objectives that are to be conveyed to their subordinates. In case some of these subordinates are TCAs, the procedure `convertToOrders` converts regional objectives to an explicit set of launching orders. As policy 3 shows, SCAs receive information from their colleagues, `sensorNet` and `weaponsNet`.

Policy 3: SCA(id,myColleagues,myRMAs,sensorNet,weaponsNet)

Initialization
Get addresses of `myColleagues, mySubs, sensorNet, weaponsNet`
Input: Information from `sensorNet,weaponsNet`, colleagues, informants, etc.
Output: Directives to subordinate commanders
Data: mode
while *recv(Es,`sensorNet`,self,now,$\vec{z_s}$) or recv(Ew,`weaponsNet`,self,now,$\vec{z_w}$) or recv(E,`myColleagues`,self,now,$\vec{z}$) or recv(E,`mySubs`,self,now,$\vec{z}$)* **do**
  computeMode($\vec{z_s}$,$\vec{z_w}$,now,newMode)
  computeToColleaguess($\vec{z}$,now)
  send(info,self,`myColleagues`,$\vec{z}$,now)
  if *mode≠newMode* **then**
    switchMode(newMode)
    send(chgMode,self,`mySubs`,mode,$\vec{z}$,now)
    if *mode=war* **then**
      send(chgMode,self,`mySubs`,$\vec{z}$,now)
      compRegionalObjectives($\overrightarrow{region_i}$, $\overrightarrow{obj_i}$)
      for *i=1,n* **do**
        if *isRMA($region_i$)* **then**
          send(regionalObjective,self,$region_1$, obj_i,now)
        else
          convertTolaunchOrder($region_i$)
          send(launch,self,$region_1$, obj_i,now)

4. COMPOSING C2 STRUCTURES

This section describes how to create composite C2 structures that have been proposed for BDM policies. They are inherently hierarchical. For example, a SCA is at the top with a set of RCAs immediately below them. Each RCA has a set of TCAs. In addition, all of them read the `sensorNet` and the `weaponsNet` to obtain sensory cues and operational status of weapon systems. If sensors or weapons are under their command, they update these nets accordingly.

4.1 Command Structures

In order to model the described C2 models, we define syntactic structures. They consist of trees of command agents. We then describe an example scenario and show how to computationally assess the advantage and disdavantage of each of these structures.

DEFINITION 5 (C2 STRUCTURES). *A command structure is a 7-tuple (ID,tcaID,rmaIc,scaID,tcaSehema,rcaSchema,scaSchema) where ID is a finite set of identifiers of all the entities of the model. rcaID, scaID ans tcaID are the identities of the TCAs, RCAs and SCAs. Additionally, ID includes two special identifiers `senosrNet` and `weaponsNet` satisfying the following conditions.*

1. *ID=tcaID∪rmaID∪scaID∪{`sensorNet`, `weaponsNet`} where sets on the right hand side are disjoint.*
2. *tcaSchema, rcaSchema and scaSchema are respectively sets of* well typed *(to be explaind shortly) instances of the following:*
   TCA(id, mySuperiors,sensorNet,weaponsNet),
   RCA(id,mySuperiors, myTCAs,sensorNet,weaponsNet),
   SCA(id,myColleagues,myRMAs,sensorNet,weaponsNet).
   The well typedness *of the schema instances is defined as follows:*
   (a) *All instances of `myColleagues` in scaSchema instances are subsets of scaID.*
   (b) *All `subOrdinate` instances in scaSchema are subsets of rcaID∪tcaID, and all `subOrdinate` instance of rcsSchema are subsets of tmaID.*
   (c) *All `superior` instances in tcsSchema are singleton subsets of rcaID∪tcaID, and all `superior` instances of rcaSchema are singleton subsets of scaID.*
   (d) *Suppose id, id'∈ ID are identifiers of two schema instances, say `XcaInstance` and `YcaInstance`, chosen from tcaSet, rcaSet or scaSet. Then, id∈`mySuperior` set of `XcaInstance` iff id'∈`mySubordinates` of `YcaInstance`.*

Lemma 1 states some simple conditions satisfied by C2 structures.

LEMMA 1 (C2 STRUCTURES). *C2 structures satisfy the following conditions.*

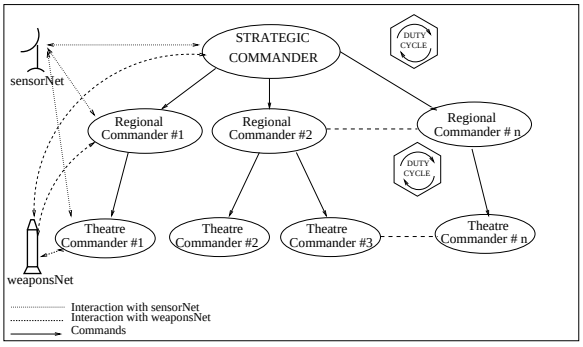


Figure 1: A Hierarchical C2 Tree

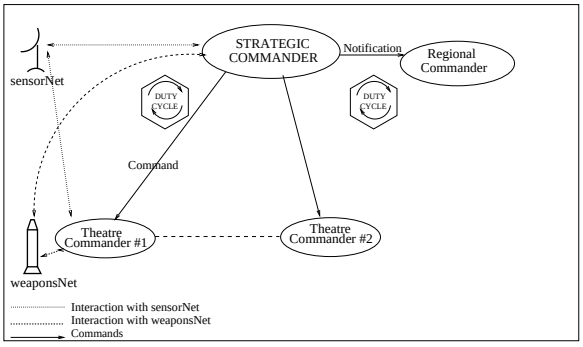


Figure 2: A Partially Flattened C2 Tree

1. *Every C2 structure is a forest of trees.*

2. *Every tree in a C2 structure has at most 3 levels, in which every path from a root to a leaf lists the agents in the order [SCA>RCA>TCA]*

We omit the trivial proof of Lemma 1, and give three example C2 structures.

4.2 Example C2 Structures

We specify three C2 structures that have been proposed for BMD. The first one shown in figure 1 has a SCA at the top with two RCAs directly below. Every RCA has two TCAs each. To specify a C2 structure, we use the ID space {0,1,2, 11,12,21,22}, where the SCA is identified as 0, and the two RCAs are 1 and 2 and their TCAs are identified respectively as 11, 12, and 21,22. Thus the schema that specify this structure is formally described as the septuple (ID,tcaID,rcaID,scaID, tcaSchema,rcaSchema,scaSchema) where the components are given as follows:

1. ID={0,1,2,11,12,21,22,sensorNet,weaponsNet}

2. scaID={0}, rcaID={1,2}, tcaID={11,12,21,22}.

3. scaSchema={SCA(0,∅,{1,2},sensorNet,weaponsNet)}

4. rcaSchema={RCA(1,{0},{11,12},sensorNet,weaponsNet), RCA(2,{0},{21,22},sensorNet,weaponsNet)}

5. tcaSchema={TCA(11,{1},sensorNet,weaponsNet), TCA(12,{1},sensorNet,weaponsNet), TCA(11,{1},sensorNet,weaponsNet), TCA(12,{1},sensorNet,weaponsNet)}.

Figure 2 shows a structure in which the RCAs are eliminated and therfore the SCA directly commands the TCAs. Its formal definiton is obtained by using the following choice of the septuple (ID,tcaID,rcaID,scaID, rcaSchema,rcaSchema, scaSshema).

1. ID={0,1,2,sensorNet,weaponsNet}

2. scaID={0},rcaID=∅, tcaID={1,2}.

3. scaSchema={SCA(0,∅,{1,2},sensorNet,weaponsNet)}

4. tcaSchema={TCA(1,{0},sensorNet,weaponsNet), TCA(2,{0},sensorNet,weaponsNet)}

The third example given in figure 3 shows a structure in which the TCAs autonomously address BMD threats. Its formal definiton is obtained by using the following choice of the septuple (ID,tcaID,rcaID,scaID,rcaSchema,rcaSchema, scaSchema).

1. ID={0,1,2,sensorNet,weaponsNet}

2. scaID=rcaID=∅, tcaID={1,2}.

3. tcaSchema={TCA(1,∅,sensorNet,weaponsNet), TCA(2,∅,sensorNet,weaponsNet)}

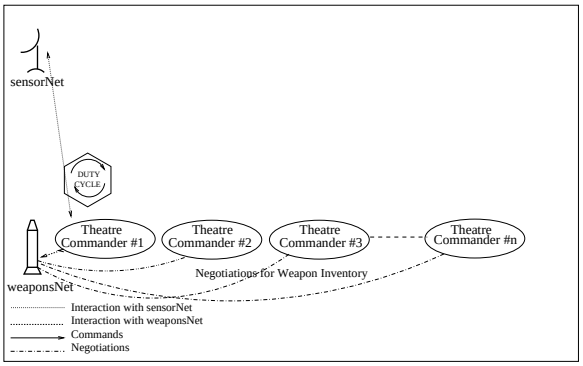


Figure 3: A Flattened C2 Tree

5. METRICS FOR C2 STRUCTURES

The *quality* of a C2 structue depends upon the application specific measures of effectiveness (MoE) and measures of performance (MoP). For a single weapon, we use *measure of effectiveness*, as the former is considered as a product of *availability*, *reliability* and its *capability*, and *timing* where the first two are expressed as probabilities.

Because our model is not intended to discriminate one kind of weapon from another, we do not distinguish between traditional measures of individual weapon's effectiveness and performances based on its hardware characteristics. But we take them as given and develop measures of overall system effectiveness and perforamance that can distinguish potential tradeoffs between organizational policies. In developing such a methodology, we propose to metrize the loss of fidelity suffered in going from a hierarchical C2 strcuture to a flattened

Function Call	Symbolic Delay
exec(withinHit,$L_{self,j}, Lt_{self,j}$)	t_{wHit}
assignTargetToMB($L_{self,j}, Lt_{self,j}$)	$t_{aTarget}$
cancelable($m_{self,j}$)	$t_{evalCan}$
delayed($m_{self,j}$)	$t_{delayed}$
holdable($m_{self,j}$)	$t_{evalHold}$
exec(setHoldTimer,$m_{self,j}, L_{self,j}, Lt_{self,j}$)	t_{timer}
exec(lock,$m_{self,j}, L_{self,j}, Lt_{self,j}$)	t_{lock}
exec(cancel,$m_{self,j}, L_{self,j}, Lt_{self,j}$)	t_{cancl}
exec(reLaunch,$m_{self,j}, L_{self,j}, Lt_{self,j}$)	t_{reL}
identifyMBA($C_{self,j}, m_{self,j}$)	t_{idM}
send(), recv()	t_{com}
Interruptible launch time	t_{int}
non-interruptible launch time	t_{nonInt}

Table 1: TCA Performance

Case	Time
Launching	$n.[t_{aTarget} + 2.t_{comm} + t_{wHit} + t_{lock}]$
Cancelling	$n.[t_{id} + t_{can} + t_{evalCan} + t_{com}]$
Holding	$n.[t_{id} + t_{hold} + t_{evalHold} + t_{com} + t_{timer}]$
reLaunching	$t_{reL} + t_{com} + t_{delayed}$
timer	$t_{reL} + t_{com}$

Table 2: Computing TCA Cycle Times

Function	Time
computeBattlePlan($\vec{z}, \vec{w}$,canAchieve)	t_{plan}
computeFiringMBAandTime($\vec{w}$, $\overrightarrow{(TCA_i, \vec{L_{i,j}}, \vec{t_{i,j}})}$)	t_{fPlan}
computeHoldingMBAandTime($\vec{w}$, $\overrightarrow{(TCA_i, \vec{H_{i,j}}, \vec{t_{i,j}})}$)	t_{hPlan}
computeCancellMBAandTime($\vec{w}$, $\overrightarrow{(TCA_i, \vec{C_{i,j}}, \vec{t_{i,j}})}$)	t_{cPlan}
cancellBattlePlan()	t_{cancel}

Table 3: RCA Performance

one, vs. the time saving obtained due to a lower communication cost. This can be compared with the QoS metrics that measure the individual stream fidelity vs. inter-stream synchronization in multimedia systems [15, 16].

5.1 Metrics for TCAs

We consider the following individual metrics for TCAs.

Availability: Availability measures the chances that a weapon is available in a launchable condition at the requested time. Suppose that a TCA has n weapons available out of a a total of N. Then we say that $\frac{n}{N}$ is the *instantaneous availability* of a single weapon for deployment. The running average of this ratio over a specified number of duty cycles is taken as the measure of *availability*.

Reliability: This is metrized as three likelihood values. That of the reliability of the process of launching, canceling and holding fire as three decimal values in the interval [0,1] referred to as D_l, D_c and D_h.

Capability: Capability is defined as the ability to reach a target. Suppose `cap`$(x, y, z, x_h, y_h, z_h, t)$ is the likelihood that a target at (x, y, z) hits the target at (x_h, y_h, z_h) (again relative coordinates) within the time t. All spacial coordinates have their origins at the TCA.

Timing: The timing characteristics of TCAs are dependent upon our description of the duty cycle. We use policy 1 to describe these. In doing so, we assume that the duty cycle is not put on hold when a launch, delay or cancel command is issued (i.e. the calls are asynchronous in operating system's terminology). Independent of the time it takes to call for a launch, we assume that a launch command takes $t_{int} + t_{nonInt}$ time to complete and any cancel or delay order must be received within t_{int} time.

Table 1 shows the performance numbers required to compute the cycle time of a TCA. It has delays for some major functions such as assigning launches to missile batteries, initiating, cancelling and delaying launches etc., that are duties of a TCA. Based on the value in table 1, we now calculate the period of a TCA duty cycle. Going back to policy 1, the initilization cost is borne only at the begining, and therefore does not affect its cycle time. The duty cycle is specified in the `while` loop in policy 1. Therefore, because all paths through the `switch` statement have to receive an event and output data to `weaponsNet` and `sensorNet`, they add an overhead of $3.t_{comm}$. Thus the finer analysis of the time to cycle now depends upon the options taken in the duty cycle is given in table 2.

Based on the values in table 1, the maximum and the minimum cycle times of TCA agents are the maximas and minimas of the values in the right hand column of table 2. We refer to them as t_{min}^{TCA} and t_{max}^{TCA}. One of the consequences of our agents is that any command reaching a TCA agent will be delayed by [t_{min}^{TCA} and t_{max}^{TCA}] time interval before it is considered by the agent, and will be acted upon a within the same time interval.

5.2 Metrics for RCAs

One of the main functions of a RCA is to coordinate between the various TCAs under its command. Therefore, the loss of a TCA would result in the loss of such coordination. If the objective is to attack one threat missile that is destroyed by a single launch, this would not matter. But if the regional objective is to destroy a sequence of missiles, and the coordination function is not taken over by either a SCA or another TCA (in which case they act as a RCA), then there would be a loss of coordination. We are currently working towards developing a metric to quantify the effect of coordination. But, given our design of the RCA duty cycle, we can measure the time it takes to coordinate between different TCAs.

Table 3 shows the runtimes used to compute SCA delays. Based on these delays, we can compute the cycle time taken for RCAs to function. The maximum and the minimum of these values are respectively the maximum and the minimum of $\{t_{switch}, t_{fPlan} + t_{hPlan} + t_{cPlan} + 3.n.t_{com}\}$. We denote them by t_{min}^{RCA} and t_{max}^{RCA} respectively.

5.3 Metrics for SCAs

SCAs are at the highest level of the C2 hierarchy. As stated

Function	Time
computeMode($\vec{z}$, $\vec{w}$,now,newMode)	t_{mode}
computeToColleagues($\vec{z}$,now)	t_{coll}
computeRegionalObjective($\overrightarrow{(region_i, obj_i)}$)	t_{rObj}
convertToLaunchOrder($region_i$)	$t_{convert}$

Table 4: SCA Performance Numbers

before, their duty cycle consists of exchanging information with colleagues, reading inputs from the `sensorNet` and `weaponsNet` and change its mode and compute regional objectives, based on the inputs. Table 4 show symbolic values that are used to compute the period of the SCA duty cycle. Using the data from table 4, the cycle time for a SCA falls within the time interval $[t_{min}^{SCA}, t_{max}^{SCA}]$ where t_{min} is $2t_{com}$ + t_{mode} + t_{coll} and t_{max} is t_{min} + t_{swMode} + t_{rObj} for a SCA that does not act as a regional commander in a partially flattened C2 structure. Conversely, the latter case adds another delay of $m.t_{convert}$ to the maximum where the SCA acts for m number of RCAs. We can use these delay estimates to compute many delays due to C2 hierarchies and structures, as shown in the next section. That would require communication delays.

5.4 Delays due to C2 Structures

Now we use the computed delays for proposed agents and estimates of communication delays to compute performance characteristics of hierarchical, partly flattened and totally flatted C2 structures. In order to do so, we assume that any sender-to receiver communication takes a maximum of t_{net} delay. This includes all communications from the sensors and TCAs to `sensorNet` and `weaponsNet` respectively and between and two agents. As has been observed earlier, the communication delay - although cannot be bounded above by a uniform value t_{net} - results in different agent systems having different degrees of awareness over the C2 domain. We first compute them as follows.

Hierarchical Structure: In the hierarchical structure, a sensory information issued (simultaneously) by one or more sensors take $2.t_{net}$ time to reach the SCA because it goes via the `sensorNet`. Because the hierarchical structure shown in figure 1 has three levels, a total communication delay of $4.t_{net}$ is encountered. However, each agent can take between $t_{min}^{(T/R/S)CA}$ and $2.t_{max}^{(T/R/S)CA}$ time to make its decision. The total time taken between a sensor receiving a cue and a missile being ordered to be launched is between $4.t_{net} + t_{min}^{SCA} + t_{min}^{RCA} + t_{min}^{TCA}$ and $4.t_{net} + 2.t_{max}^{SCA} + 2.t_{max}^{RCA} + 2.t_{max}^{TCA}$. The launching delay of $t_{int} + t_{unInt}$ needs to be added to this in order for a missile to leave the launching platform.

Partially Flattened Structure: In a partially flattened structure, as the one shown in figure 2, the SCA is directly connected to its TCAs. Then, based on the explanation given above, the time difference between receiving a cue and the missile being ordered to be launched is between a minimum of $3.t_{net} + t_{min}^{SCA} m.t_{convert} + t_{min}^{TCA}$ and a maximum of $3.t_{net} + 2.t_{max}^{SCA} 2.m.t_{convert} + 2.t_{max}^{TCA}$. As expected the computation delay associated with the RCA is reduced, with an increase in the SCA decision making time in order to convert regional battle objectives to launching schedules. Again, the launching delay is $t_{int} + t_{unInt}$. As noted earlier, the time advantage comes at the cost of the loss of inter-regional planning and coordination.

Flattened Structure: In a fully flattened structure as the one shown in figure 3, the TCAs act on their own. Therefore the time delay between a sensory cue being issued by a sensor and the time of ordering a missile is between a minimum of $2.t_{net} + t_{min}^{TCA}$ and a maximum of $2.t_{net} + 2.t_{max}^{TCA}$. The launching delay of $t_{int} + t_{unInt}$ still needs to be added to this number in order for the missile to leave the launching pad.

5.5 Computing MoEs and MoPs

As stated earlier, the C2 domain uses many measures of effectiveness and performance. But, upto now, we have computed end-to-end delays only. We now visit some of those that were described in section 1.

The three measures of effectiveness we considered in section 1 were that (1) Did an interceptor kill the threat missile, (2) Did a follow up kill the threat missile and (3) Did the C2 structure hold the fire when asked to? The three measures of effectiveness that were considered are (4) Can the interceptor destroy the missile at a safe height? (5) Did the battle manager assign a weapon in time to destroy the threat? (6) Can the BMD system identify a threat? We now show how our analysis can help answer the first five of these questions with the help of other the metrics described in section 5.1.

The first question has two aspects. The first is that is there sufficient time to hit the target missile? That can be answered with the help of our timing analysis. Suppose that time between sensors receiving the cues of the threat and the interceptor missile leaving the launching pad, as calculated in section 5.4 is between $t_{min}^{?CA}$ and $t_{max}^{?CA}$. Then, depending upon the continuous cues received by the sensors, we can compute the projected path of the threat missile. Given the position and flight characteristics of the threat missile and its position on the computed path at the time of launching the counter missile, equations of motion can be formed and solved in order to determine if they meet at some time. Furthermore the computed height of the intersection point can be used to determine the distance of impact to friendly territory in order to determine if any humans are at danger from secondary debries - thereby being able to compute an answer to the first measure of performance, re-stated as (4) in the previous paragraph. The second aspect of the first question is the probability of the interceptor being able to destroy the threat missile. This is a hardware characteristic of the TCA that needs to be given - that we have listed as *reliability*.

The second effectiveness metric has two aspects. Firstly, an RCA can ask for two interceptor missiles to be scheduled to be launched with sufficient time lag so that if the first fails to destroy the threat then the second can be released and canceled otherwise. Alternatively, the RCA(s) can keep generating requests to destroy an airborne missile throughout its flight. The maximum number of times this call can be safely repeated can be computed using our timing analysis. Again the probability of destruction is a characteristic of interceptor reliability.

The third effectiveness measure is again a combination of

timing and reliability. Our timing analysis can be used to compute the time to call for a launch and then in a later duty cycle to call for a cancellation. Conversely, the ability to cancel depends upon the times t_{int} and t_{nonInt} of the missile battery and its reliability value of cancellation.

The fourth (i.e. the first measure of performance) was addressed earlier. As shown it can be answered using our timing analysis.

The fifth (i.e. the second measure of performance), the question as to if the battle manager assigned a interceptor in time is purely a matter of timing analysis and predictability of the threat missile's path and other dynamics of flight. We have sketched as to how this can be addressed. The last measure of performance, that of detecting a friend from a foe is beyond the scope of system policy analysis.

6. RELATED WORK

Michael et al. [8] developed an iterative approach for studying the timing constraints of a prototype ballistic missile defense system by using models expressed in UML for Real-time extension (UML-RT), which are then translated into coarse-grained simulation models that are exercised using the OMNeT++ simulation engine. The integration of the UML-RT models with simulation models provides a seamless process for rapidly constructing executable prototypes for the purpose of analyzing timing constraints and deriving system requirements from them. The effectiveness of the approach was demonstrated for the sensor-netting capability of a missile defense system.

Singaraju and Borky [11] created a methodology for evaluating force-structure alternatives against operational tasks. They developed a set of four measures of effectiveness for comparing alternative structures: (i) *operational effectivess*, the ability of the force structure to address current and projected tasking; (ii) *affordability*, the ability of the proposed alternative to fit within reasonable budget projections; (iii) *technical risk*, availability of the required enabling technologies and products to implement the system or systems involved on a given schedule; and *integration*, the ability of the proposed alternative to maintain continuity of services to warfighters and to fit into an evolving force structure in addition to backward compatibility. Specific elements of operational effectivenss used within the methodology include

Response time: The ability to meet service delivery time requirements of customers, including the time to generate mission tasking and the latency in delivery.

Coverage: rate and continuity of information collections, delivery of service, and other service to warfighters.

Operability/Supportability: ability to deliver services reliabily with acceptable requirements for staffing, useable equipment and other infrastructure.

Information quality: sensor resolution, communications data transmission and error rates, spatio-temporal positioning accuracy, correctness in identifying targets, robustness against counter-measures, and similar factors.

Effect delivery quality: ability to induce desired and controlled effects across the spectrum from denial to destruction and against all targets of interest, including projectile, directed energy, materil delivery, jamming and other means.

In contrast to our approach, the approach proposed by Singaraju and Borky involves conducting cursory qualitative assessments of the effectiveness of alternative force structures – force structure encompasses much more that just command and control.

Simpkins, Paulo, and Whitaker [10] developed a methodology for validating the outputs from simulation models such as Wargame 2000; Wargame 2000, developed by the U.S. Department of Defense, and is used to analyze command and control (C2) structures, and in particular, human interaction (e.g., commanders and their staff) in defending against ballistic missile threats. Their methodology for validating missile-defense simulation models, such as Wargame 2000, is to define a set of measures of effectiveness or performance regarding command and control and then compare the values for the measures of effectiveness or performance against those run in an independent model: if the models differ in terms of measured values, then this indicates that the validity of the models need to be further investigated.

Combs, Parry, and Towery [4] developed a methodology for evaluating the performance of Commander-in-Chief (CINC) staff support in conducting warfighting tasks (e.g.,. planning, revising plans, monitoring situation awareness information) in a computer-generated decision environment. They defined methods to measure the effectiveness of carrying out the task from one level of the command hierarchy to another (i.e. from supported to supporting units within the hierarchy).

Similarly, Rothrock [9] uses computer-based simulation to assess the performance of human decisionmakers–either individual operators or teams – in their interaction with a ballistic missile defense system. The *abstraction hierarchy* proposed by Rothrock is itself a means for systematically deriving measures of effectiveness and performance based on the command structure and other characteristics of the ballistic missile defense system.

White, Young, and Kelsch [14] investigated the tradeoffs of different system architectures to support situational awareness. They model the *situational awareness architecture* as a hierarchical communication systems architecture, along with a *tactical Internet* that supports dissemination of situational awareness data. The tradeoff analysis can be used to derive requirements for the command structure, or given a particular command structure, determine whether measures of effectiveness or performance can be met.

Athans [2] discusses the need to have a uniform, agent-based distributed hybrid control system to manage battles that involve a multitude of air, ground, space and sea based defenses. He describes detailed models a of *naval battle group (BG)* that consists of a carrier accompanied by escorting platforms consisting of ships, aircraft and submarines. The control structure of the battle group is hierarchical: headed by a *composite commander (CWC)* that delegate responsibilities to *anti-submarine warfare commander (ASWC)*, *anti-surface warfare commander (ASUWC)*, *anti-air warfare commander (AAWC)* that manage and respond to the multitude of sensory information received by the battle group. He also describes the functions of a BMD system that detetcts threats, track targets, descriminates between decoys and weapons and manage damage assesments provided by orbiting and ground-based sensor systems. Based on these inputs, this multisensor information must be fused to map weapons to targets and control engage-

ment functions.

Numerous contributions have been made in the area of agent systems. The work we are most familiar with is described in the textbook by Subrahmanian et al. [12]. The book presents a framework that is capable of specifying agent based systems that communicate with each other by sending messages to others and reading messages from private mailboxes of themselves. Here agent systems are specified by the actions that are permitted and prohibited by specification, where some prohibitions may be removed due to massages arriving later in an agent's mailbox. The authors later extend their work to probabilistic temporal agents [5]. The type of software modules which we need to develop are substantial extensions of the logic programming paradigm of [5] in which event driven agents are fired to take actions (changes of internal database, changes in receptivity to inputs, changes in form of outputs) based on obligations, temporal constraints, probabilistic decisions, etc.

Although this paper does not formulate all the rich semantics that BMD agents demand, our ongoing work is progressing towards doing so.

7. CONCLUSIONS

Defending against ballistic missiles has been studied from many perspectives because BMD management and deployment is a task undertaken by military forces, it comes under their C2 structures. Mindful of this tradition, we have formulated BMD modeling as a distributed control problem between communicating agent system that follow their own duty cycles eenforcing their individual policies that, taken together, contribute to the end goal of destroying threat missiles before they cause damage. We do so by formulating the doctrine and policies of our agents using ECA rules. Our agents exchange information relating to sensor cues and weapons status to two virtual entities referred to as `sensorNet` and `weaponsNet`. We show how to construct traditional C2 structures using our agents.

Based on timing estimates to perform each task, we show how to specify the period of duty cycles of our agents. Using these estimates and communication delays, we show how to compute times taken by a specified C2 structure between a cue received at a sensor about a potential threat and the firing of an interceptor. We sketch how these estimates can be used to compute some traditional measures of performance and measures of effectiveness.

Our work-in-progress addresses defining MoE and MoP estimate at the higher levels of traditional C2 hierarchies. We are also formulating our agent systems as communicating distributed hybrid systems that enforce a distributed policy base. We intend to investigate the effects of communication failure, and the discords caused by various attacks and impersonations on our distributed agent system. Our eventual goal is to show how MoE and MoP decorated higher level policies can be refined to and enforced by a low-level communicating agent system.

We are also working towards extracting policy-compliant strategies for BMD C2 by specifying doctrines and policies as Prolog rules. Then for a given attack, a plan (strategy) for decision makers to follow which, when executed, always produces a successful response with intercepts within a prescribed period from launch can be extracted as successful answer substitutions provided by the Prolog rule engine. Conversely, if a plan (a strategy for decision makers) is given, the same Prolog mechanism can be used to extract *weak points* of it - namely attacks for which the plan fails to provide required protection.

8. REFERENCES

[1] M. Athans. The expert team of experts approach to Command-and-Control (C2) organizations. *IEEE Control Systems Magazine*, pages 30–38, Sept. 1982.

[2] M. Athans. Command and Control (C2) theory: A challenge to control science. *IEEE Trans. Automatic Control*, AC-32(4):286–293, 1987.

[3] D. C. Caffall and J. B. Michael. System-of-systems design for the Ballistic Missile Defense System. In M. Wirsing, A. Knapp, and S. Balsamo, editors, *Radical Innovations of Software and Systems Engineering in the Future*, volume 2941 of *Lecture Notes in Computer Science*, pages 108–121, Berlin, 2004. Springer-Verlag.

[4] R. Combs, S. Parry, and C. Towery. Development and implementation of measures of effectiveness for the Universal Joint Task List in the joint theater level simulation. In C. Alexopoulos, K. Kang, W. R. Lilegdon, and D. Goldsman, editors, *Proc. Twenty-seventh Winter Simulation Conf.*, volume 1, pages 1139–1146, Arlington, Va., 1995. IEEE.

[5] Juergen Dix, Sarit Kraus, and V.S. Subrahmanian. Heterogenous temporal probabilistic agents. *to appear in ACM Transactions of Computational Logic*, 2005. avilable at: http://www.acm.org/pubs/tocl/accepted.html.

[6] R. L. Garwin. Holes in the missile shield. *Sci. Amer.*, pages 70–79, Nov. 2004.

[7] J. B. Michael, P. E. Pace, M.-T. Shing, M. Tummala, J. Babbitt, M. Miklaski, and D. Weller. Test and evaluation of the Ballistic Missile Defense System. Technical Report NPS-CS-03-007, Naval Postgraduate School, Sept. 2003.

[8] J. B. Michael, M.-T. Shing, M. H. Miklaski, and J. D. Babbitt. Modeling and simulation of system-of-systems timing constraints with UML-RT and OMNeT++. In *Proc. Fifteenth Int. Workshop on Rapid System Prototyping*, pages 202–209, Geneva, Switzerland, June 2004. IEEE.

[9] L. Rothrock. Developing performance metrics in pre-deployment systems using the abstraction hierarchy. In M. J. Chinni, editor, *Proc. Military, Government and Aerospace Simulation Symposium*, volume 1, pages 97–101, Washington, D.C., Apr. 2000. Society for Computer Simulation Int.

[10] S. D. Simpkins, E. P. Paulo, and L. R. Whitaker. Case study in modeling and simulation validation methodology. In B. A. Peters, J. S. Smith, D. J. Medeiros, and M. W. Rohrer, editors, *Proc. Winter Simulation Conf.*, volume 1, pages 758–766, Arlington, Va., Dec. 2001. IEEE.

[11] B. K. Singaraju and J. M. Borky. A methodology for evaluating architectural solutions. In *Proc. Aerospace Conf.*, volume 6, pages 485–494, Big Sky, Mont., Mar. 2000. IEEE.

[12] V.S. Subrahmanian, P. Bonatti, J. Dix, S. Kraus T. Eiter,

F. Ozcan, and R. Ross. *Heterogeneous Agent Systems*. MIT Press, June 2000.

[13] D. B. Weller, D. C. Boger, and J. B. Michael. Command structure of the Ballistic Missile Defense System. In M. J. Savoie, H.-W. Chu, J. Michael, and P. Pace, editors, *Proc. Int. Conf. on Computing, Communications and Control Technologies*, pages 42–48, Austin, Tex., Aug. 2004. Int. Inst. of Informatics and Systemics.

[14] D. A. White, L. Young, and G. R. Kelsch. Modeling and simulation of situational awareness in the tactical Internet. In *Proc. Military Communications Conf.*, volume 3, pages 872–876, Boston, Mass., Oct. 1998. IEEE.

[15] Duminda Wijesekera and Jaideep Srivastava. Quality of service (qos) metrics for continuous media. *Multimedia Tools and Applications*, 3(2):127–166, 1996.

[16] Duminda Wijesekera, Jaideep Srivastava, Anil Nerode, and Mark Foresti. Experimental evaluation of loss perception in continuous media. *ACM Multimedia Systems*, 7(6):486–499, 1999.

BMD Agents: An Agent-Based Framework to Model Ballistic Missile Defense Strategies

Duminda Wijesekera[1], J. Bret Michael[2] and Anil Nerode[3]
George Mason University[1],
Naval Postgraduate School[2] and
Cornell University[3]

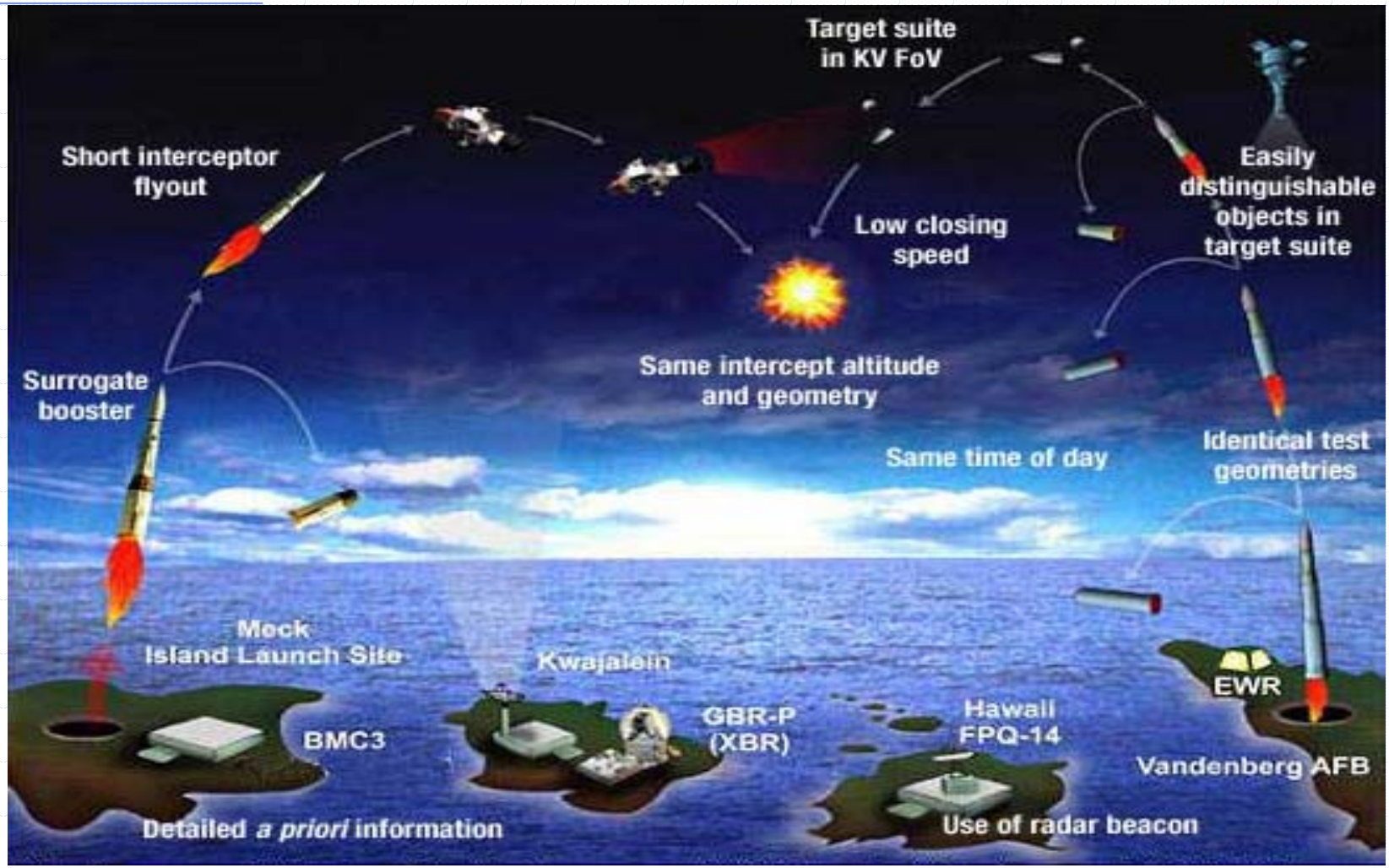
Disclaimer

- The views and conclusions contained in this presentation are those of the author and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of the U.S. Government.

Acknowledgements

- This work is sponsored by the Missile Defense Agency

The environment



http://www.ucsusa.org/global_security/bmd/bmd_test.html

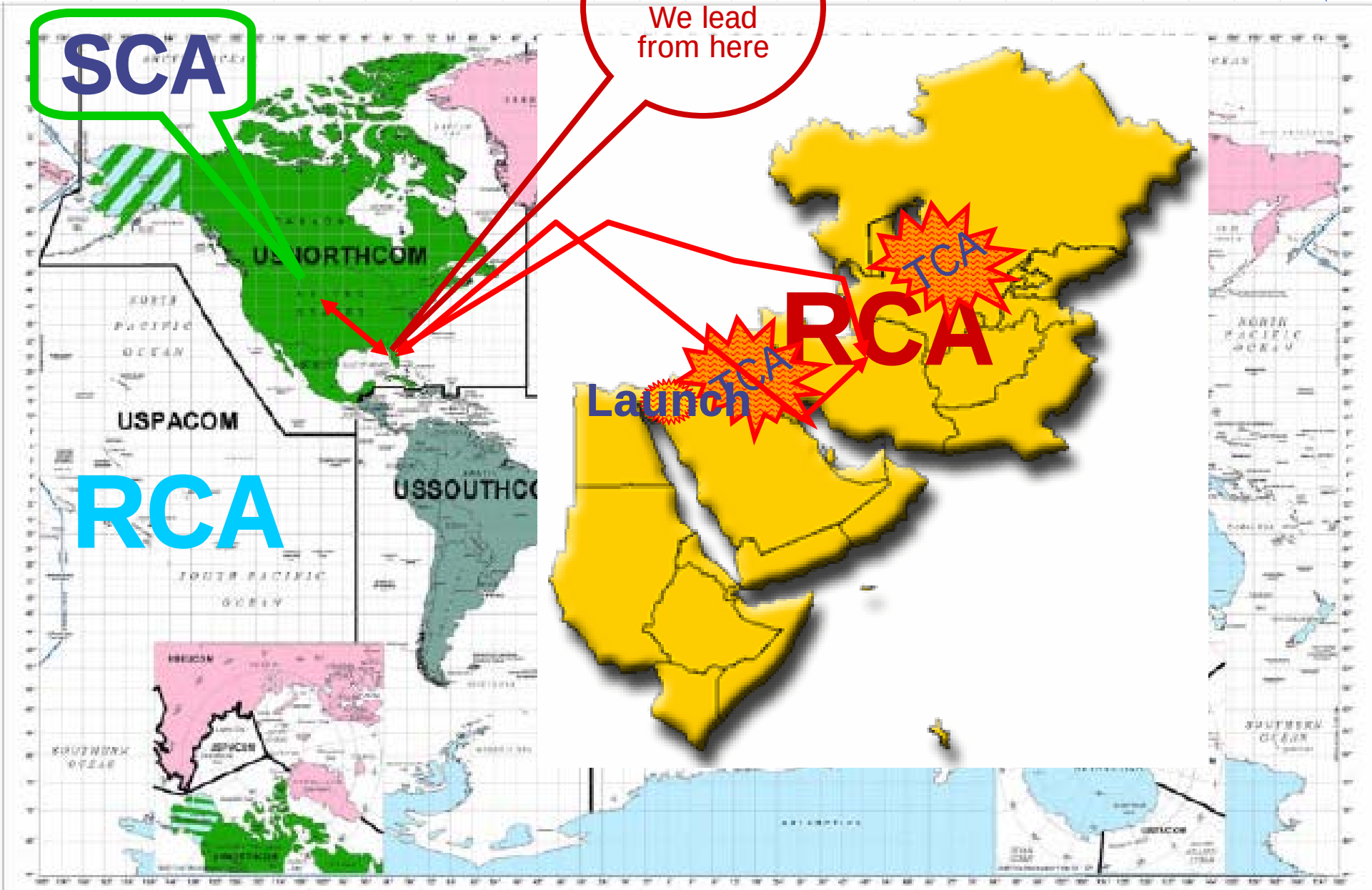
BMD modeling requirements

- Must account for
  - Deliberate planning
  - Crisis action planning
- Individuals follow a *kill chain*
  - Map the individual duties to agents
- Optimize QoS Measures
  - MoP: Measure of Performance
  - MoE: Measure of Effectiveness

Modeling choice

- Use a collection of agents based on the roles they play in the missile defense environment
  - Strategic Command Agents
    - Directs high-level strategies among many regions
  - Regional Command Agents
    - Coordinates regions consisting of multiple theaters
  - Theater Command Agents
    - Directs theater-level actions

In pictures

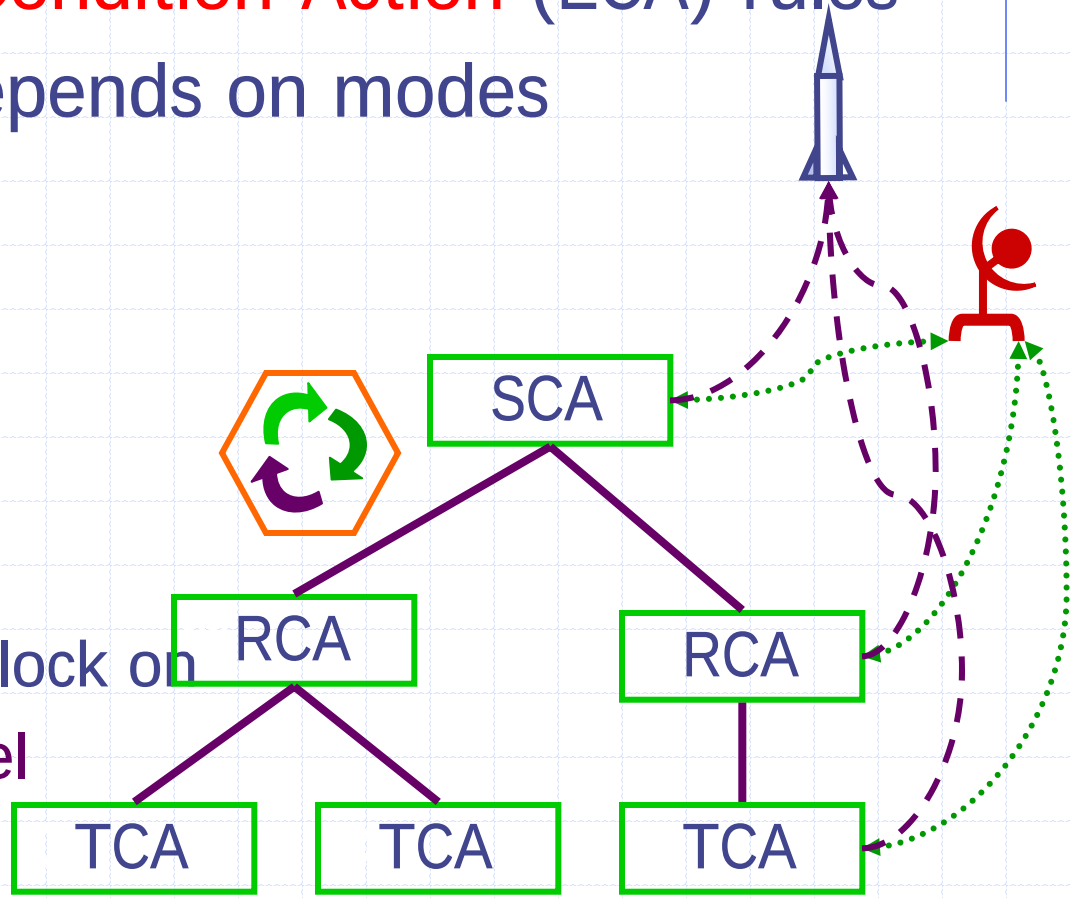

SCA
SCA
We lead
from here
USPACOM
USSOUTHCO
RCA
TCA
RCA
TCA
Launch

The operating environment

- SensorNet
  - Information gathered (using sensors) about flying objects of interests are broadcasted here
- WeaponsNet
  - Operational status about weapons systems are broadcasted here

Modeling details

- Build using Event-Condition-Action (ECA) rules
- Agents behavior depends on modes
  - Peacetime
  - Pre-hostilities
  - Hostilities
  - Post hostilities
- Duty cycle
  - Acquire target and lock on
  - Launch, wait, cancel
  - Assess



Designing agents 1: SCAs

- Obtain information from
  - SensorNet, WeaponsNet, AND *friends,* and
- Assigns tasks with timing constraints to subordinates consisting of
  - Changing modes (peace, war, pre-war, post-war)
  - Computing regional objectives of tracking, and destroying flying objects
  - Altering and/or canceling current objectives
- Informs *friends* as necessary

Designing agents 2: RCAs

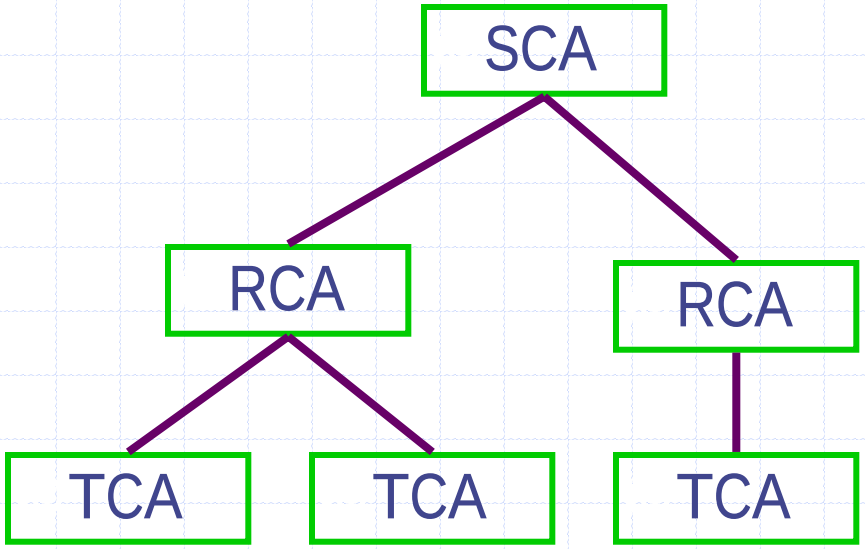
- On receiving directives from superiors
  - Get data from SensorNet, WeaponsNet and
- Assign time-constrained tasks to TCAs consisting of
  - Pass on changing mode commands (war, pre-war, post-war) to subordinates, and change own mode.
  - Computing regional firing, holding (fire) and canceling fire orders and assign them to TCAs
- Send feedback acknowledgements (about their ability to comply with orders) to superiors

Designing agents 3: TCAs

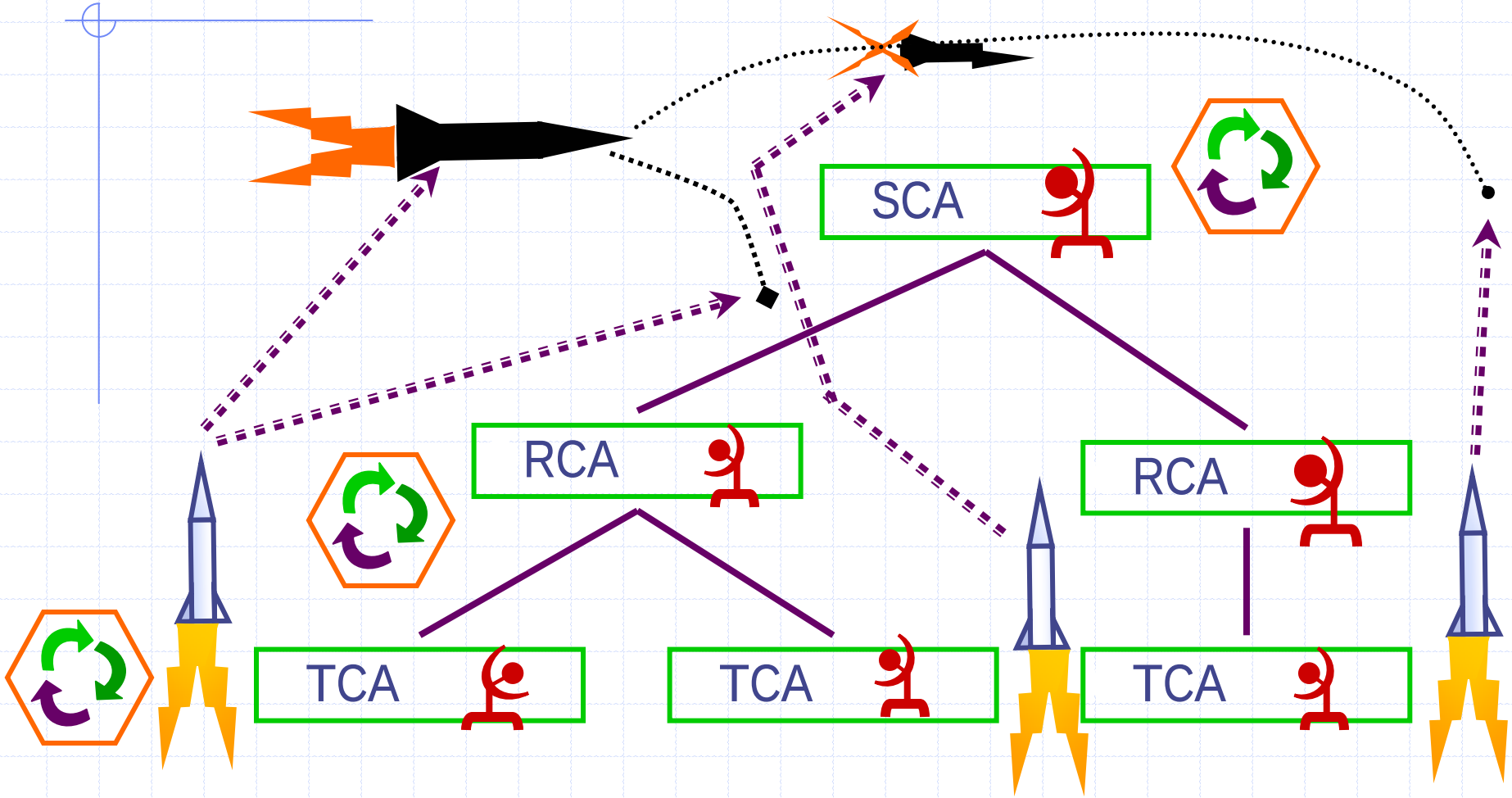
- On receiving directives from superiors
  - Get data from SensorNet, WeaponsNet and
  - Change mode on command (war, pre-war, post-war)
  - Execute the duty cycle of acquiring/locking on/firing/ assessing damage to the target
  - On command, recompute firing/reload/holdfire/cancel schedules per weapon under own command
- Send feedback acknowledgements (about their ability to comply with orders) to superiors
- Inform sensorNet an weaponsNet about changes to tracked targets and weapons status

Designing agent communities

- Need to design command, control and communication (C^3I) structure for agents to model BMD functionality
- Use real-life examples

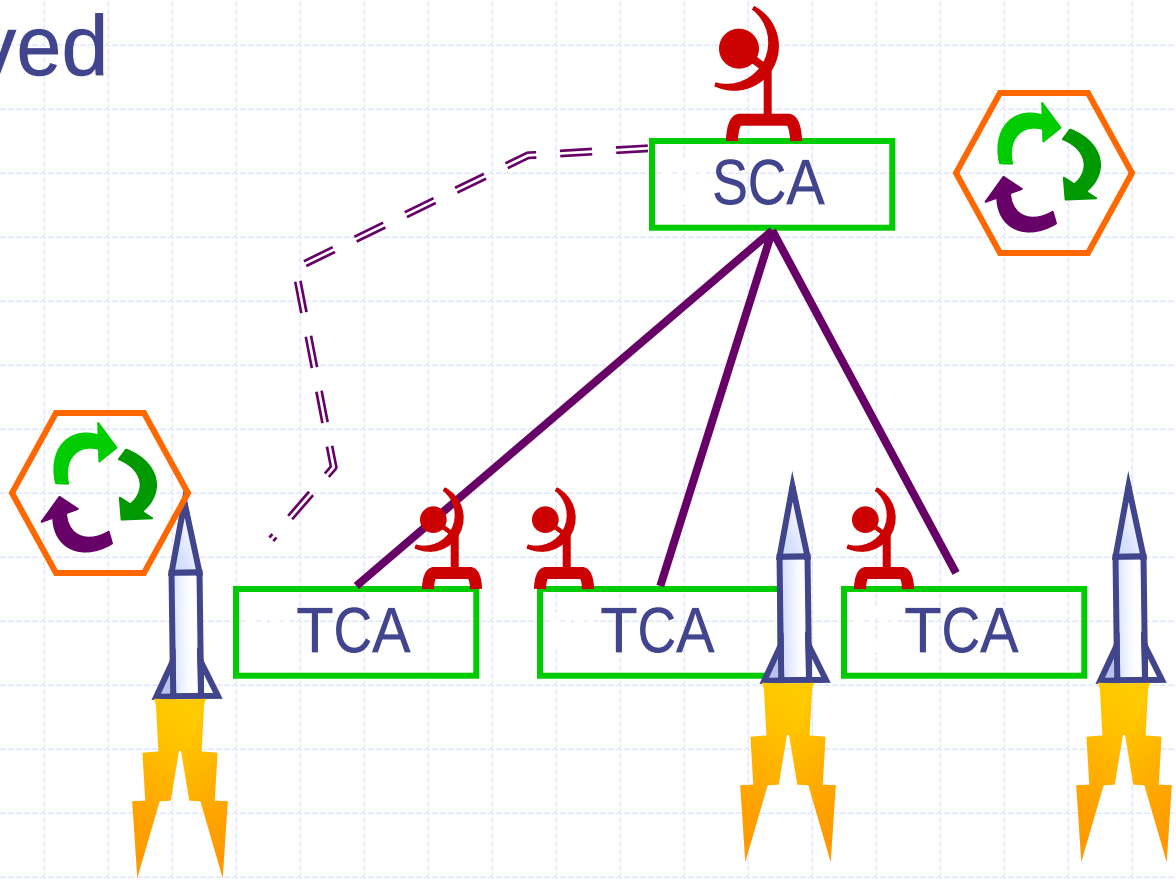


Command structure 1: Hierarchical



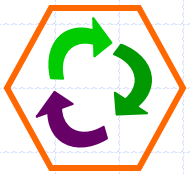
Command structure 2: Partially flattened

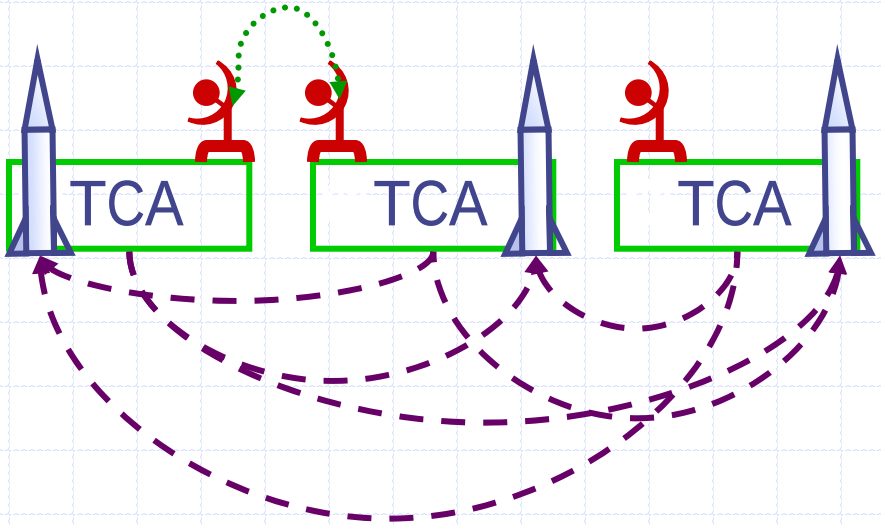
- RCAs removed



Command structure 3: Flattened

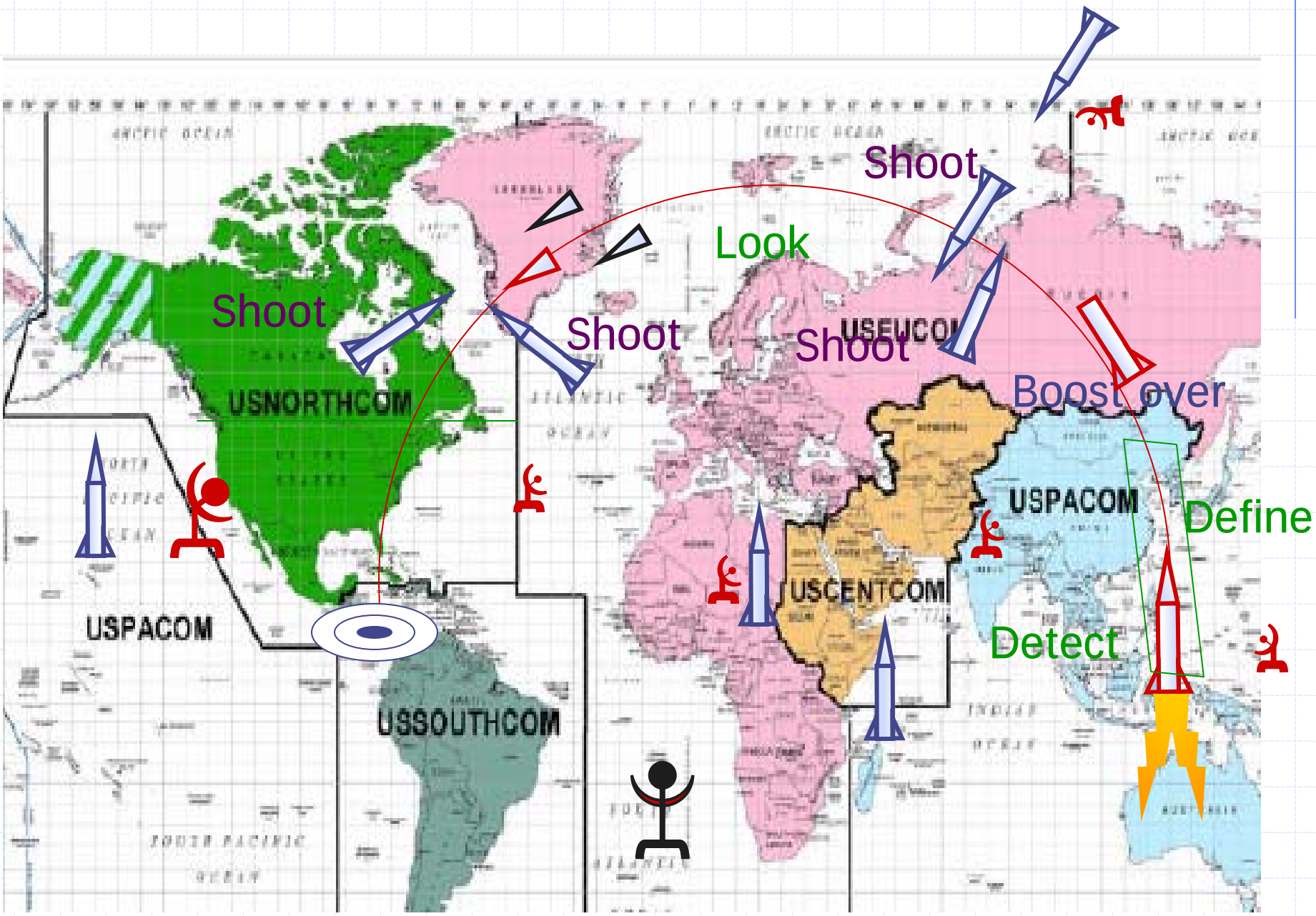
- TCAs work autonomously





Composing agents: C^2 structures

- A tree consisting of at most 3 levels
- Every level has at most *one type* of agents
- Agents listed in the SCA/RCA/TCA order
- Every agent knows its superiors/subordinates
- Every SCA knows all of its *friends*
- Lemma: A simple static analysis algorithm can detect if any collection of agents is a C2 structure
- Limitation: Does not account for duty polymorphism (i.e. SCA's doing RCA's work)


Shoot
Look
Shoot
Shoot
Shoot
Boost over
Define
Detect
USNORTHCOM
USEUCOM
USPACOM
USCENTCOM
USPACOM
USSOUTHCOM

Analysis objectives

- Can the treat missiles be destroyed before it hits or scatters debris over intended target?
  - Missiles entering airspace need to be identified and categorized as threat, potential threat, or benign
  - Targets and travel trajectories/times be computed and all fragments tracked and destroyed in threat missiles
  - Commanders need to obtain authority to aim at missiles
    - This authority need to propagate through the command chain
    - Takes time to lock on and fire
    - Do follow-up shots destroy the threat missile?
    - If object is reclassified as benign, need to cancel/delay firing

Preliminary results

- Compute periods for duty cycles of agents using
  - Worst-case estimates for command execution times
  - Performance delays of weapon systems
- Compute command propagation times through statically-composed C2 structures using
  - Worse-case communication delays
  - Computed duty cycle periods
- Using these estimates, one can compute if a properly identified threat missile can be intercepted with a particular weapon

Limitations

- Need to account for
  - Hit/destroy probabilities
  - Reclassification of missile status and the ability to recall/re-target missiles
- Need to incorporate measures

Measures

- Measures of Effectiveness, such as
  - Can launches (or repeat launches) destroy threat missiles?
  - Does the system hold fire if missile status is reclassified?
- Measures of Performance, such as
  - How much above ground are they destroyed?
  - Delay in reacting to reclassification

Related work

- Many approaches
  - Force-structure-based
  - Strategy-based
- Some examples:
  - Athens: C2 Theory, *IEEE Trans. on Automatic Control* 32, 4 (1987), pp. 286-293
  - Michael, Pace, Shin, Tummala, Weller, Miklakski, Babbit: Test and evaluation of BMD systems, NPS TR-CS-03-007, 2003
  - Garwin: A hole in the missile shield, *Scientific American*, 2004, pp. 70-79

Summary

- Presented a preliminary ECA rule-based agent framework to capture BMD C2 requirements where
  - Strategy and policy are written as BMD rules
- A preliminary formulation of a well-formed agent society for BMD C2
- A back-of-the-envelope timing calculation

Ongoing work

- Experimenting with a model that uses probabilistic temporal reasoning
  - (Probabilistic Temporal Agents of Kraus et al.)
- Using rules to code policies and strategies
- Formulating a framework for both
  - Hierarchically building the MoEs and MoPs
  - Computing
    - Probability of achieving the numbers
    - Schedules for launches